Rich Denney

5th International Conference on Software Quality (5ICSC)

October 23-26,1995

Stouffer Renaissance Hotel
Austin, Texas

Program Committee

✓ **Joyce Statz**, TeraQuest Metrics (chair)
Anita Carleton, Software Engineering Institute
David Card, Software Productivity Solutions
Bill Curtis, TeraQuest Metrics
Taz Daughtrey, Babcock and Wilcox
Dennis Frailey, Texas Instruments
Warren Harrison, Portland State University
Watts Humphrey, Software Engineering Institute
Sue McGrath, SAS
Bob Yacobellis, Motorola
Richard Zultner, Zultner and Company

Paper Reviewers

Acosta,Ramon D.
Amann,Vaughn
Beckwith,Sue
Biggs, Mel
Boger, Allen
Bond, Hollis
Boudreaux, Charles
Brune,Kevin D.
Burton, Kim
Card, David
Clark, Mickey
Curtis, Bill
Darrah, Jim
Daughtrey, Taz
Demirors, Ondr
✓ Denney, Richard
Depwe, Robert
Evans, Kyle
Flories, Kathy
Folsom, William T.
Forbes, Bill
Frailey, Dennis
Fulford, Robin Ann
Futrell, Bob
Garza, Paul
Glantzberg, Hughes
Harrison, Warren
Hines, Mary Lou
Holub, Kirk
Ibrahim, Linda
Johnson, Allen
Kleen, Linda
Lamia, Walter
Lanoux, Judy
LeVie, Jr., Donn
Marks, Peter
Martin, Cecil
Martin, Tracy
McDonald, Ryan
McGrath, Sue
Perkins, Rusty
Perkins, Michael T.
Petersen, Don
Phipps, David L.
Ramachandran, Ram
Reed, Russ
Rightmer, Jerry
Ring, Dennis
Rising, Linda
Samuel, J.
Shier, David
✓ Streun, Geree
Trujillo, Rod
Werth, Laurie
Yacobellis, Bob
Zoric, Maureen

“Return on Investment from Continuous Software Process Improvements in an International Company”

Peter Lloyd
Manager, Software Quality Assurance
Schlumberger Austin Product Center
Austin, TX

Return on Investment from Continuous Software Process Improvements in an International Company.

P.M.Lloyd, Schlumberger Austin Systems Center, Austin, Texas, 78720-0015. Lloyd@austin.asc.slb.com

Abstract

In 1990, Schlumberger committed to continuous software quality improvement using the Software Engineering Institute's Capability Maturity Model. A functionally complex, multi-million line software product, developed at different sites around the world, and critical to our core business is used as a case study. Process improvement initiatives over the last 5 years are mapped to increased development output and external measures of product quality. Return on investment for various process improvements is quantified, and twelve best practices described, plus some failures. Other business benefits of continuous process improvement are summarized.

Contents:

1. Introduction

Schlumberger's Oilfield Services business units operate in over 100 countries and employ 30,000 people of as many nationalities. They provide a complete range of seismic, drilling, well evaluation, well completion and field development services, which include data acquisition, transmission, processing and interpretation. Software for integrated petrophysical, geological and geophysical analysis is also licensed to our customers.

The Austin center provides software systems for 13 other engineering centers around the globe which build tools and write applications software. Austin is also responsible for assuring the smooth integration of applications for several product lines. For example, consistency is provided across applications by assuring tools built at different sites can be run in combination, and the data transmitted and interpreted on various customer platforms.

Engineering support to the different sites also includes maintenance of the Oilfield Services software configuration management system, defect tracking data base and common software library. The software process improvement program is also managed from Austin. This effort involves the training of 600+ software development engineers across all 14 sites, and coordination of Quality Assurance activities (which include alpha testing of some integrated product lines).

In 1990, Austin committed to continuous process improvement using the SEI multiphase approach to process maturity and process management [1, 2]. A description of the corporate wide effort, which cites early examples from this site, is given by Wohlwend and Rosenbaum [3]. By mid 1992, QA groups had been set up at all the 14 Oifield Services engineering centers with strong cooperation and resource sharing between each.

The goal of this paper is to look in more detail at the return on investment from the continuous process improvement effort over the last 5 years, and to identify the most effective 'best practices' which have developed, and continue to evolve.

2. Measuring Process Improvements and Returns

Five elements are critical to our management of engineering activities.

a. Each center has an Engineering Plan, which commits to deliverables over the next 12-18 months, and provides further vision of the direction of product development. The plans for all 14 sites are made in careful coordination between the Research and Engineering, Operations and Marketing groups, and are updated every 6 months.

b. Monthly reports, detailing project progress to a granularity of 3-5 person project teams, are distributed between each center, marketing and operations. They allow actuals to be carefully tracked against plan.

c. Monthly resource loading and spending statements are made at a similar level of granularity to track costs.

d. Configuration management and defect tracking systems that are networked between the different centers allow engineers to share code and to track problem reports across organizations.

e. An annual review and appraisal cycle is conducted where individual and team efforts over the last year are analyzed, and goals are set in place for development and to measure improvement over the next twelve months.

This infrastructure provides a sound basis on which to build a software quality system. It also provides the foundation for a rich historical database to track project progress, identify delays or missing deliverables, the cost of such process breakdowns, and to recognize successes (and associated financial savings).

Since late 1989, independent measurement and detailed analysis of the development process has been made, and specific software quality improvement initiatives undertaken. It is therefore also possible to map process improvements to productivity levels, and external measures of product quality.

3. The Study

The results of such an analysis are now made for one of the integrated product lines that is managed in Austin but which has application development at 5 different sites. From 1990-1994 (the period of this study) between 25-30 engineers were involved in its development and testing at Austin. Incrementally, about 250,000 lines of new executable C and C++ code were added annually. This is in addition to assuring systems integration of an equivalent amount of applications code by a similar number of engineers at the other sites.

4. Historical Data

An SEI assessment at the end of 1989 rated the effort in Austin at Level 1, quartile 2 (20-40% of Level 2 requirements being met). Many of the symptoms of a chaotic development effort were to be seen. Despite managers, developers and testers working extremely hard to get the release out (weekends worked, vacations postponed, stress levels high) the project deliverable was 3 months late and operations (our customer) was hardly satisfied with it. They complained it was buggy, difficult to use, and some functionality they had implicitly expected was missing.

There was a strong belief by developers and management that things could be done better, and there was a firm commitment by both management and software developers to do so.

Year	*Improvement*
1990	independent quality assurance with alpha test responsibility.
	project planning with phased release of new functionality.
1991	following a documented development process (and support docs).
	defined requirements (and improved developer and alpha testing)
	monthly project tracking sheets used on all project lines.
1992	quantifiable acceptance criteria and stress test scenarios before commercialization.
	formal technical reviews and revision of some process guidelines.
	post mortems, and QIP's (quality improvement plans) in managers' goals.
1993	improved requirements gathering and analysis.
	development of component tests for system functionality.
	causal analysis of bugs found during beta test and risky release reports.
1994	intercenter product definition/development teams.
	intercenter training with focus on communicating and sharing best practices.
	ISO 9001 certification.

Table 1: Summary of major process improvement activities each year

4.1: 1990

In 1990, an independent quality assurance function with responsibility for alpha testing was introduced. Project planning methods were also carefully analyzed and a phased release process to introduce new functionality was used for the first time. This was a change from the previously used 'big bang' approach, when all new functionality was integrated in the same timeframe. A potential cost overrun of 7% in the annual project budget (caused by delays to the previous year's major delivery) was made up with these improvements; albeit in part by ongoing heroics by the development and test teams. The 1990 system and applications entered beta test closer to schedule (though still a month late) and with improved functionality and quality. The latter was measured by the percentage of defects in the data base which were missed by development and alpha testing, but were reported by operations during beta testing. This percentage which 'slipped through' to operations, went down from 28% to 15%. An internal, though independent, assessment rated the group at L1-Q3, a visible improvement over the previous year.

4.2: 1991

In 1991 software development was conducted under recently defined Software Process Guidelines. The SPG had been written by software practitioners, and efforts were made to assure buy-in across the center. It adopted a waterfall model, and described discrete deliverables which had to be signed off technically and by management at each phase of the process. There were several support documents (templates, how to 'cookbooks', C coding guidelines) introduced in conjunction with it. This, and more systematic and consistent use of monthly project tracking sheets, provided greater project visibility to managers and also resulted in improved requirements analysis and product definition. That in turn translated into improved coordination with operations and better testing by both the developer and alpha testing teams.

In 1991, the new development came in on schedule, and this made up for a potential 5% project cost overrun caused by the sustaining team, still trying to fix problems from the previous years' release. Quality was up: only 10% of defects in the data base slipped through to the field in beta test. An end of year post mortem and SEI assessment concluded that a repeatable process was essentially in place.

Year	Process Maturity	Improvement on previous year	% Bugs in DB reported in beta	ROI as % of budget
1990	L1-Q3	one quartile	28	7
1991	L2-Q1	two quartiles	15	5
1992	L2-Q3	two quartiles	9	6
1993	L3-Q1	two quartiles	10	10
1994	L3/ISO 9001		7	7

Table 2: Process Maturity Improvement Vs Product Improvement

4.3: 1992

In 1992 the sustaining effort (managed for the first time by a well defined team within the group) ran to plan, and quantifiable acceptance criteria were set for the first fully commercial release. They included a 2 day scripted 'GO/NOGO stress test scenario' which simulated extreme commercial use of the product, and had to be completed flawlessly. A requirement for maximum allowable bug severity in external releases was also introduced, with severity quantified using a scoring system called PEG's. With PEG'ing, the defect is rated against three criteria, each on a defined scale of 0-10.

(i) P is for the 'Pain' or hardship inflicted when encountering the problem, measured by lost time or rework.

(ii) E is for the degree of adverse 'Effect' on the quality or correctness of the product.

(iii) The Frequency of occurrence expected from a non-expert user; but PEG sounded 'catchier' than PEF, so the name stuck!

Formal Technical Reviews were also introduced for the new development cycle (with help from Daniel Freedman) and statistics collected to measure their ROI. They showed an estimated 6% improvement on development efficiency due to early defect detection.

Unfortunately, a poorly planned/risked, non level 2 activity during the new development cycle resulted in extra unplanned effort which wiped out the productivity gains from the introduction of FTR's. However, the 1992 deliverables were met, and quality levels compared well with the previous product release. An independent assessment measured the effort as now being strong Level 2. Post mortems (held in a non threatening, non adversarial environment) were used to evaluate what could be done better in the next development cycle. Having continuous quality improvement goals in the objectives of the managers and their teams, had now become a part of the culture.

4.4: 1993

In 1993 the sustaining effort overran by 4% of the product lines' annual budget due to a conscious decision by the development team to react to some late requirements by our operations group to leverage off new hardware peripherals to increase user efficiency. A porting effort was managed concurrently with the sustaining and new development work. Despite the broader and more challenging goal of running on different hardware platforms, both the new functionality and ported code were released early to beta test. At the same time, tighter quality acceptance criteria (maximum allowable PEG ratings) were applied. The early deliveries represented an overall cost saving of 10% against project budget. Of this, FTR's again contributed a 6% saving on project costs.

Note how cost savings have been cumulative. Improvements one year fold into the next, so savings are ongoing. The sum of net savings from 1990, '91, '92 and '93 amounts to about 30% of the product budget. They more or less covered the project costs of the additional, but highly successful, porting effort in 1993.

For the software developed in 1993, fewer than 7% of defects in the database slipped through to beta. This was less than 200 bugs compared with 500 which slipped through to beta 3 years previously. Some Level 3 activities (first introduced in 1992 as pilots) started to become part of the culture. These included more detailed requirements analysis, improved testing at component, integration and alpha test levels, and increased automation of those tests. Better leverage was also made from statistics generated by our configuration management system, with the introduction risky release reports to focus high return areas for testing. More detailed analyses of why bugs slipped through to beta, and what could be done to eliminate them, or catch them faster, were made.

An independent assessment at the end of 1993 indicated that the project team was following a defined process.

4.5: 1994

In 1994, the software was commercialized 5 months ahead of that achieved in '92 and '93, with the same strict acceptance criteria.

A new system was also developed within the group to broaden the applications scope from 5 to 8 centers. It used, almost exclusively, common software (from our own reuse library) or commercially available components. The first release made its schedule, and our customer application groups have been very positive; not always the experience with 'first systems releases'. A big part of the success was that monthly product definition team meetings were held between the different groups. This allowed conflicting needs of the different groups to be quickly identified, and trade-offs to be made by the technical staff. Note that the successful reuse of common software libraries and commercially available software increases the development team's output by an order of magnitude, when compared with building the software from scratch.

Efficiencies in three other areas were also noted in 1994 as a consequence of evolving a shared and common process between the 14 different sites.

Staff transfers have become increasingly easy to manage between departments and different centers. There is no long start-up cycle as new tools or methods have to be learned, because we do things in a similar way at all 14 sites.

Training in core competencies of Software Engineering also reached a record level. With all centers increasingly following a similar development process and sharing best practices, it has been increasingly easy and cost effective to manage training classes, and tailor material for local applications. In 1994, classes were given to as many software engineers in core competencies as in the previous four years, with a significantly lower training budget actually being used. Clearly the success leveraged strongly off efforts during those previous years.

ISO 9001 certification (to the 9000-3 software guideline) was also more straightforward than we had been led to expect from various industry reports. Having a visible, defined process across the Austin center proved easy for our auditors to assess, and allowed certification to be achieved just 6 months after our operations group requested it. There was only a 0.3% project overhead in costs and additional training. The scope of the certification "Design, development, integration and production of software and hardware products for Oilfield Services" illustrates the importance of strong intergroup coordination in the distributed engineering environment we are working in today.

Savings on training and on the straightforward ISO certification (when compared with costs at less mature sites for which we have data) corresponded to about 7% of overall project costs in '94. The cumulative 30% efficiency gain reported for 1993, appears to have been carried over into 1994 (with the new systems development successfully completed in parallel with a sustaining and porting cycle).

No SEI assessment was done in 1994, but the process has certainly grown stronger, because ISO 9001 auditing has helped put extra emphasis on process compliance and visibility. It has also involved more staff in the overall scope of quality activities (i.e. personnel and purchasing, previously not too involved in CMM driven improvement activities).

5. Return on Investment

Each year about 2-3% of the engineering budget is invested in process improvements, part of an overall commitment to independent quality assurance (including alpha testing) of 8%.

While each year the 5-10% measurable return for the 2 or 3 new practices put in place is not staggering, the improvements would appear to be both effective and cumulative. This is evidenced by the development group now managing new development, sustaining and major porting projects concurrently by well defined teams within the group.

In addition, the analysis has not attempted to quantify the additional business benefits to marketing of having products released on schedule with required functionality, and to operations and their customers of having low levels of defects.

6. Were Improvements Inevitable?

One might argue that as the project got older and the team more experienced with the software, that product quality and development efficiency would improve naturally.

This is not supported by tracking size, bug count and development history of a similar product line developed during the '80's (indeed, with some of the same team members involved in the case history described above). As the product became bigger, and more complex, the defect rate seen by operations (open SDRs in figure 1) climbed more steeply than code size itself. This suggests that development improvements due to increased experience can be canceled out by increased complexity of the overall system and supported applications. When a concerted effort was made to address the testing in 1987 and to improve the development process in 1989, the number of open defects started to drop dramatically (despite continued increase in software size). The software is currently under maintenance and still in commercial use. There are now fewer than 100 reported 'open' defects in the 1 million lines of assembler code.

It was the results of these late 1980 improvements which helped get management and practitioner buy-in to the SEI model, at a time when there was real concern by some engineers that a better process was fine, but would mean unacceptably low productivity levels.

If one believes in the degree of entropy (chaos) having a natural tendency to increase as projects become bigger, and the needs they satisfy become more complex, then the 5-10% annual ROI described above is all the more impressive.

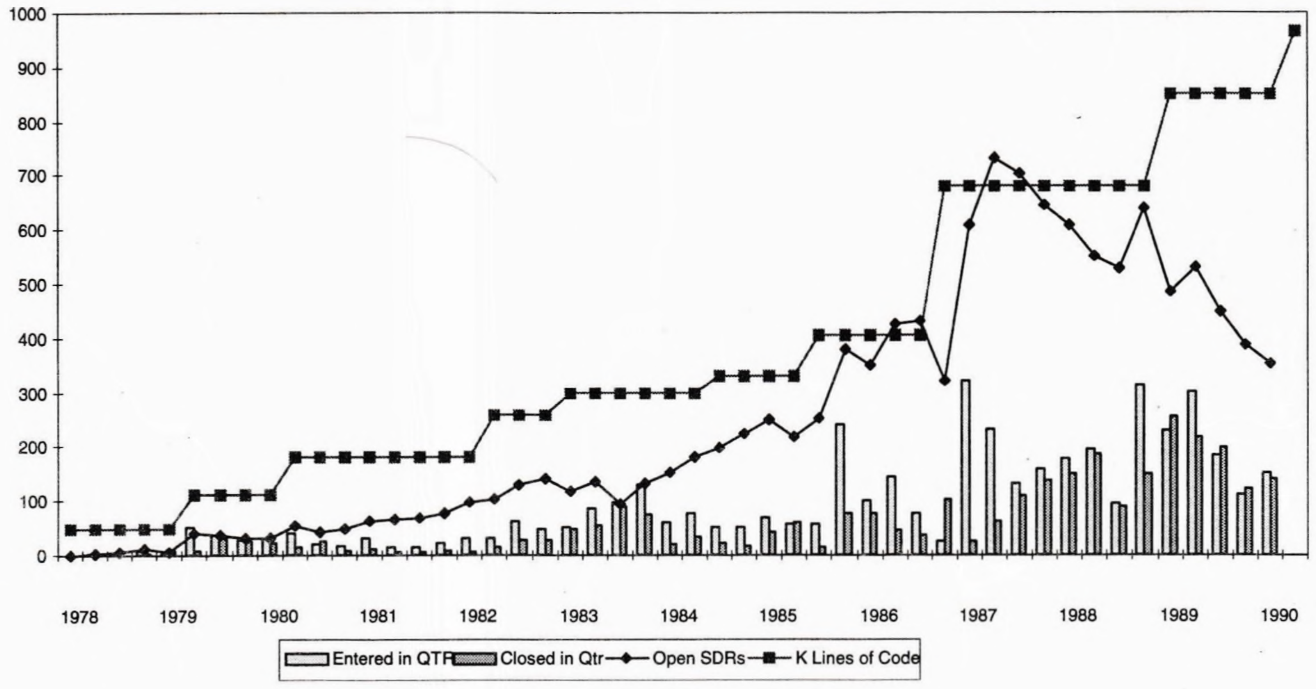

Figure 1 Plot of LOC vs open defect reports during the '80s for previous similar product line.

7. Summary of Best Practices

The SEI-CMM (Software Engineering Institute's Capability Maturity Model) promises that as you reach a Defined Process (Level 3) you can begin to analyze and streamline the development process.

During 1994 we have seen some good examples of new practices evolving and old practices improving, which are managing to cut overhead, improve communication and make us more efficient. These include the following:

1. Product Definition/Development Teams (PDT's) representing developers and their 'customers' at different engineering centers, in operations and marketing, are meeting monthly to review product status. The meetings provide a regular focus to prioritize needs, balance tradeoffs, and manage late changes. There is increased product ownership by all interested parties, with key decisions increasingly being made by consensus at the technical level.

2. Assuring that the specifications have measurable QA acceptance criteria and are explicitly testable (with testability/usability scenarios) is a must. Local QA groups at each of the different centers will help achieve that goal. It is often a good idea to include in the same document a description of customer requirements, a scenario of how the product will be used, and an engineering specification of what will be delivered to meet those needs (with the development plan as an appendix). This helps cut paperwork, and lends tractability.

3. Reporting project progress in a consistent and graphical manner, recording effort, size and product quality has an important impact in improving intergroup communication. It will also provide a sound database when planning estimates need to be made for future projects.

4. Catching potential defects early in the development cycle (where they are cheaper to address) continues to show a strong return on investment. The use of formal technical reviews to help do this has spread to all Oilfield Centers.

5. Transferring ownership of the Component Test plan from the development team to QA, when the functionality is frozen, and alpha testing is moving smoothly helps avoid needless test plan duplication and keeps plans up to date. If the developers wish to go back and do maintenance work on a component, they have an up to date test plan to run themselves before the new QA hand-off.

6. A 'risky release' report can help focus integration test efforts. The relative risk of releases (submissions of new or modified code to the configuration management system) is assessed as a function of the number of lines changed, components needing to be relinked due to such changes in other parts of the code and hours testing prior to release. These are standard outputs of the configuration manager's report after a build, so the listing can be generated automatically. It allows those releases which are most likely to have problems to be tested first, and problems reported quickly. ✓

7. Automating Test Plans, and using automated regression test scripts adds tremendously to integration and QA test efficiency. It has allowed target builds for external systems releases to be turned around in 24 hours, with 100% systems and 90% application test coverage.

8. The use of PEG's (see section 4.3) communicates in a quantitative manner the seriousness of defects from a user's perspective. This helps focus bug fix efforts during alpha test and maintenance cycles. Maximum allowable PEG ratings are now used (in conjunction with 'stress job scenarios') to set quality acceptance criteria for software releases for most product lines.

9. Having easy access to critical lifecycle documents is a key element in project/process visibility, and communication. Using WWW as a project book layout offers increased accessibility to: requirements, specification, design and test documents, minutes of PDT and launch meetings, FTR issues and statistics, defect tracking status, development plans and monthly status reports.

10. Periodic causal analysis meetings are recommended to analyze bugs and determine how they could have been caught earlier in the development process by changes to current peer review, design or testing practices. Implementing such changes results in ongoing product quality and process improvements.

11. Post mortem meetings, internal SEI assessments and ISO audits also help identify weak parts of the process, where action can increase product quality or development efficiency. Action items can be put in the goals of project managers and department heads to assure commitment to improved continuous quality improvement.

12. ISO 9001 internal audits have proven effective in assuring project compliance with the defined process. Certification is quick and painless when a well defined software development process is in place, which is well understood, has strong developer and management ownership and which has high visibility. It allows the organization to capitalize on the software improvement efforts over the years in the commercial marketplace.

Identifying best practices at each of the Oilfield Services Engineering Centers, communicating them, and sharing local successes, have been goals for the various QA groups. Technology transfer has been accomplished by quarterly QA managers meetings, using joint SEI or ISO assessment teams at each center, by a concerted training effort, and having training or process support documents easily accessible in our on-line work environment. We also communicate with the QA staff in other organizations to share good ideas, practices and training material.

8.0 Conclusions: What worked well and what didn't

In retrospect, despite not really understanding what was supposed to happen in the rarefied atmosphere of Levels 4 and 5, key players were convinced back in 1990 that a better process gives better products. There was real commitment to make it work amongst developers and management. And while there were concerns voiced about loss of creativity and too much paperwork, the creative, low paperworld environment we had experienced in the late '80s was not that great either.

A commitment by management is important, but a belief by the practitioners themselves is critical, because they play the ultimate role in creating the peer pressure to make changes happen and then consolidate them. Put simply, commitment must be bottom up as well as top down.

Teams of 3-4 committed and proactive individuals with detailed knowledge of the systems they are developing, or how customers will use it, appear more effective in driving process improvements than bigger committees with less specific objectives. Quality Assurance staff should play a hands-on and proactive role in the lifecycle of the products they are supporting by offering strong domain or software engineering expertise.

Accept that the ROI from any one improvement will not be staggering. Introducing 2-3 new practices a year yielded a relatively modest 5-10% annual return. However, after 2-3 years of continuous improvement activity the payback became quite impressive. The development team was 30% more effective by the fourth year of the effort.

Do not try and fix more than 2 or 3 areas of the process in any year. Expect more or less immediate returns from the areas you do address (in terms of quality or productivity improvement).

There have been some failures. A code analysis and size/complexity metrics program was put in place which adopted the approach of measuring a large number of code and project characteristics and then analyzing the data. A rather expensive CASE tool was bought to help do this with a resource commitment of 12 staff months. The results were found to be 'interesting'. However, they tended to confirm what we thought we already knew, and they provided little direction as to what to do next. Needless to say, we did not follow some of the above advice with that project. We have since adopted the more focused GQM (Goal, Question, Measure) model with metrics, and this is proving more effective. The approach here is to do root cause analysis of specific problems, and use carefully tailored measurements to track them.

Do not try to implement Level 3 (and above) review, measurement and automation practices while still operating in a chaotic, Level 1, environment. Otherwise you will waste much effort reviewing chaos, measuring chaos and automating chaos.

In general we feel we have followed Watts Humphrey's good advice about managing the software process and using the SEI maturity model. It has paid off handsomely. When problems occur, they seem to be issues with our implementation, and not flaws in the fundamental model.

9. What Next?

Analyzing our process and defining continuous quality improvement plans to increase developer efficiency and product quality will remain cornerstones of our basic approach. We will continue the strategy of addressing just two or three process improvement areas per year, for any project. We will carefully measure return on investment and peer acceptance of the improvement efforts.

Emphasis will be given to understanding how to better use design methodologies to reduce rework, and leverage more from our 'common software library' and commercially available code. We will upgrade our defect tracking system and have it more closely integrated with configuration management. Increased automation of component and integration test suites (on multiple platforms) is another target.

10. Acknowledgments

Pushing process improvements in Austin has been a team effort. From a staff of 120 software developers, over a third have contributed in the formulation, review, and revision of various process guidelines, templates, and standards. I'd like to particularly thank Wes Byrne, Susan Rosenbaum, and Jorge Boria who have been involved in the management, process improvement support, and independent assessment of the project described above, and who have helped to critically review and assure the accuracy of this paper. I'd also like to note help provided by Bill Curtis, Daniel Freedman and Stan Rifkin for providing external input at critical stages of our process evolution to our management and staff. And finally, thanks to the SEI and Watts Humphrey for providing us with a road map, a way of seeing where we started from, and good directions.

11. References

[1] Watts Humphries, "Managing the Software Process", Addison Wesley Publishing Company, 1989.

[2] Mark Paulk, Bill Curtis and Mary Beth Chrissis, "Capability Maturity Model for Software", Software Engineering Institute, CMU/SEI-91-tr-24, 1991.

[3] Harvey Wohlwend and Susan Rosenbaum, "Software Improvements in an International Company", proceedings International Conference on Software Engineering, IEEE CS Press, Los Alamitos, CA., 1993.